Planning with Nondeterministic Actions and Sensing

Eyal Amir
Computer Science Division
University of California at Berkeley
eyal@cs.berkeley.edu

Abstract

Many planning problems involve nondeterministic actions - actions whose effects are not completely determined by the state of the world before the action is executed. In this paper we consider the computational complexity of planning in domains where such actions are available. We give a formal model of nondeterministic actions and sensing, together with an action language for specifying planning domains. Then, we examine the cases of complete observability, partial observability and no observability, assuming that sensing is done automatically or needs to be done explicitly. We restrict our attention to plans of tractable plan-size or depth.

We show that planning with nondeterministic actions for polynomially represented plans has computational complexity equivalent to that of planning with deterministic actions under incomplete knowledge about the initial state, if we the domains include no observability or full observability, and consider an assumption on executability of actions. If the latter takes polynomial time, then our complexity class for all these problems is Σ_2^P-complete. If the problem of checking executability of actions is NP-complete (the general case), or we allow partial observability or sensing actions, then our complexity class is Σ_3^P-complete. For plans of polynomial depth, we find that planning in nondeterministic systems with no observations is in Σ_2^P-complete, contrary to previous conjectures of PSPACE-completeness (Haslum & Jonsson 1999). We also find that planning in nondeterministic systems with full observability or partial observability (with and without sensing actions) is PSPACE-complete for polynomial-depth plans. These results point out cases where it may be useful to use encodings in boolean formulae to perform planning, and they carefully draw the distinctions between the different scenarios involved.

1 Introduction

Planning with nondeterministic actions is an increasingly important branch of reasoning about dynamic systems. Systems that have been developed include (Giunchiglia 2000; Ferraris & Giunchiglia 2000; Cimatti, Roveri, & Traverso 1998; Bertoli, Cimatti, & Roveri 2001; Bertoli *et al.* 2001). They constitute an important counter-part to probabilistic AI planning in MDPs and POMDPs (e.g., (Dean & Kanazawa 1988; Doucet *et al.* 2000)). However, there is only little understanding of the semantics of nondeterministic planning with sensing (however, see (Reiter 2001)) or the difficulty of the problem, given different assumptions.

In this paper we present a formal transition model for nondeterministic actions with sensing and an action specification language for specifying planning problems in a natural way. Our transition model and action specification language are important for understanding and developing such systems. We do not know of a transition model and action specification language (besides the situation calculus theories in the style of (McCarthy & Hayes 1969; Reiter 2001)) that can handle both nondeterministic actions and observations in the scenarios that we explore. The transition model generalizes one given by (Bertoli *et al.* 2001) for fully observable domains.

We use this semantics to provide computational complexity results for planning with nondeterministic actions and sensing, with different assumptions about the planning scenario. Our results are summarized in Table 1 at the end of the paper. There, we compare them to results obtained for deterministic actions and for actions with probabilistic effects. Our results point out cases where it may be useful to use encodings in boolean formulae to perform planning (e.g., (Cimatti, Roveri, & Traverso 1998; Ferraris & Giunchiglia 2000)). Our results complement earlier results achieved by (Eiter *et al.* 2000; Baral, Kreinovich, & Trejo 2000; Turner 2001). *Some proofs are omitted here for lack of space. They are available from the author.*

2 Sensing and Nondeterministic Actions

In this section we define the transition model and action description language that we use for nondeterministic actions, sensing actions, and automatic sensing of some features. Our model supports sequences of actions, and is not intended for concurrent actions or ramifications of actions. We take elements from different action languages, particularly $\mathcal{AR}$ (Giunchiglia, Kartha, & Lifschitz 1997) (which includes nondeterministic actions) and $\mathcal{A}_K$ (Son & Baral 2001) (which includes sensing actions) and the formal model of (Bertoli *et al.* 2001) (includes some automatic sensing and nondeterministic actions). At the same time, our language lacks some of the expressivity that is available today in some action languages, such as $\mathcal{C}$ (Giunchiglia &

Lifschitz 1998). Nonetheless, it is expressive enough for our purpose in this paper, and it may be extended to include those missing elements. Regardless, the following description is self contained and the reader is not required to know these systems or their details.

In what follows, for a set of propositional formulae, $\mathcal{X}$, $L(\mathcal{X})$ is the signature of $\mathcal{X}$, i.e., the set of propositional symbols that appear in $\mathcal{X}$. $\mathcal{L}(\mathcal{X})$ is the language of $\mathcal{X}$, i.e., the set of formulae built with $L(\mathcal{X})$.

2.1 Transition Model

A transition system is a tuple $\langle \mathcal{P}, \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$, where

- $\mathcal{P}$ is a finite set of propositional fluents;
- $\mathcal{S} \subseteq Pow(\mathcal{P})$ is the set of world states;
- $\mathcal{A}$ is a finite set of actions;
- $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is the transition relation.

The intuition for this transition system description is that $\mathcal{P}$ is the set of features that are available for us in the world, every element in $\mathcal{S}$ is a *world state* (i.e., a subset of $\mathcal{P}$, containing propositions that are true in this world state), $\mathcal{A}$ is the set of actions in the system (these may be actions that change the state of the world, sensing actions, or a combination of both) and $\mathcal{R}(\langle s, a, s' \rangle)$ means that state s' is a possible result of action a in state s.

For a transition system $\langle \mathcal{P}, \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$ we can define observations and transitions between *belief states*. A belief state is a set of states we cannot distinguish between, given the actions performed and the observations collected so far.

- $\mathcal{S}^b \subseteq Pow(\mathcal{S})$ is a set of belief states.
- $\mathcal{R}^b : \mathcal{S}^b \times \mathcal{R} \longrightarrow \mathcal{S}^b$ is a transition function for belief states.

The intuition is that $\mathcal{R}^b(\sigma, \langle s, a, s' \rangle)$ is the resulting belief state after executing a in world state s with the consequence s', starting from belief state σ. The resulting belief state, σ', includes the effects of observations that we make about the situations involved, if there are any. This allows us to model fully observable, partially observable and unobservable domains, possibly with sensing actions and automatically observed fluents.

In general, a transition system may have $\mathcal{S}$ of size $2^{|\mathcal{P}|}$, belief states of size $2^{|\mathcal{P}|}$, and belief-state space of size $2^{2^{|\mathcal{P}|}}$. Also, $\mathcal{R}$ may be of size $2^{2*|\mathcal{P}|+|\mathcal{A}|}$. This poses a difficulty for representation and reasoning with general large transition systems. Nonetheless, compact representations exist for restricted cases of such systems. The best example is deterministic systems with a fully known initial state, where belief states are only single states, $\sigma = \{s\}$. No observations are needed to find the actual world state because we start from a fully known state and advance deterministically. In such systems $\mathcal{R}^b(\{s\}, \langle s, a, s' \rangle) = \{s'\}$, for a unique s' such that $\langle s, a, s' \rangle \in \mathcal{R}$.

2.2 Action Description Language

The transition model and belief state update function given above can represent a rich set of dynamic systems. In this section we describe a specification language that defines some of those dynamic systems and that we will use in the rest of this paper. In this language we allow actions that change the world in nondeterministic ways, sensing actions, declarations about the initial state and declarations about automatically observed features.

A logical nondeterministic domain description D is a finite set of statements of the following kinds: *value propositions* of the form "**initially** F" describe the initial state; *effect propositions* of the form "a **causes** F **if** G" describe the effects of actions; *sensing propositions* of the form "a **determines** F **if** G" say that in states where G holds, action a results in the value of F being known; and *automatic observation propositions* "**observed** F" say that F is known in every state, for F and G being *state formulae* (propositional combinations of fluent names). We say that F is the *head* and G is the *tail* of those rules.

For a domain description D we define $\mathcal{P}_D$, $\mathcal{A}_D$ to be the set of propositional fluents and actions mentioned in D, respectively. The following semantics describes the way a state changes after an action:

- If, before the execution of an action a, the state formula G is true, and the domain description contains a rule "a **causes** F **if** G", then this rule is *activated*, and after the execution of action a, F becomes true.
- If for some fluent f no activated effect rule includes the fluent f in its head, this means that the execution of action a does not influence the truth value of this fluent. Therefore, f is true in the resulting state if and only if it was true in the old state.
- If an action a has a set of rules with a combined inconsistent effect F (e.g., $F = FALSE$) and that set of rules is activated in s, then there is no state that is the result of a in s (we take this to mean that a is not executable in s).
- The result of applying an action a for which no rule is activated in s is the state s (we consider the action as possible in this state, but having no impact).

The last two principles ensure that the conditions of the rules act as conditions for the action's effects (if no conditions are met, then there are no effects), and an action is not executable iff it leads to contradictory effects (e.g., if we include a rule saying that "a **causes** FALSE **if** G"). In the latter case for s and a, there is no transition tuple $\langle s, a, s' \rangle$ in $\mathcal{R}$.

Formally, for a domain description D we define a transition relation $\mathcal{R}_D(s, a, s')$ as follows.

- A fluent $f \in \mathcal{P}_D$ is *possibly affected* by action a in state s, if there is a rule "a **causes** F **if** G" in D such that G is true in s and $f \in L(F)$.
- Let $I(a, s)$ denote the set of fluents in $\mathcal{P}_D$ that are *not* possibly affected by action a in state s.
- Let $F(a, s)$ be a set of all the heads of activated effect rules in s (i.e., if "a **causes** F **if** G" is activated in s, then $F \in F(a, s)$). We consider the case of $F(a, s) = \emptyset$ (no activated effect rules) as $F(a, s) \equiv TRUE$.

- Define (recalling that world states are sets of fluents)

$$\mathcal{R}_D = \left\{ \langle s, a, s' \rangle \;\middle|\; \begin{array}{l} (s' \cap I(a,s)) = (s \cap I(a,s)) \\ \text{and } F(a,s) \text{ is true in } s' \end{array} \right\} \quad (1)$$

We explain this definition. First, inertia is applied to all fluents that do not appear in an activated rule (regardless of whether they are positive or negative in the current state).

EXAMPLE Let $F = p \vee q$, and assume that the rule "a **causes** F **if** G" is the only rule activated in state s for action a. Then, all of $\langle s, a, s_1 \rangle, \langle s, a, s_2 \rangle, \langle s, a, s_3 \rangle$, are in our transition relation $\mathcal{R}$, for s_1, s_2, s_3 such that $p \in s_1$, $q \in s_2$ and $p, q \in s_3$ (i.e., we allow nondeterminism to not only choose between the effects p, q but also possibly affect both p, q). ■

This example may seem unintuitive at first because if p, q are both true in s, then one of our resulting states is s_2 in which p is not true. This is sanctioned by our effect rule for a, which explicitly allow this effect. If we want to state inertia so that this does not happen (e.g., that this state is possible only if we started from a state in which p was already FALSE), then we need to provide explicit rules that say so. We regard this as the responsibility of the knowledge engineer or an automated process that may generate those rules for us (as in (Lin & Reiter 1994) and others).

Our choice of this semantics for nondeterminism is mainly for its simplicity, and its natural properties. It resembles the specification of a situation calculus theory after a solution to the frame problem has already been applied (Reiter 2001). There are other semantics that are used for specifying nondeterministic dynamic systems; we relate our system to those semantics in Section 4.

EXAMPLE Another example is when we have an effect rule for a with no preconditions and no effects (i.e., both $G = F = TRUE$). If this is the only rule active in state s, then the result of s is s. In comparison, if we have an effect rule for a that has $G = TRUE$ and $F = p \vee \neg p$, then there are two possible resulting states, one in which p holds and another in which $\neg p$ holds. ■

We define the belief state update function $\mathcal{R}^b_D(\sigma, \langle s, a, s' \rangle)$ for D as follows.

- If, before the execution of an action a, the state formula G is true, and the domain description contains a rule "a **determines** F **if** G", then this rule is *activated*, and after the execution of action a, the truth value of F in the resulting world state becomes known.
- Let $F^b(a, s)$ be a set of all the heads of activated sensing rules of a in s (i.e., if "a **determines** F **if** G" is activated in s, then $F \in F^b(a, s)$) together with all heads of automatic observation statements (i.e., if "**observed** F" is in D, then $F \in F^b(a, s)$).
- For a set of propositional formulae $\mathcal{F}$ and a state s, let $V(\mathcal{F}, s) = \{F \in \mathcal{F} \mid F \text{ is true in } s\}$.
- For belief state σ, and $\langle s, a, s' \rangle \in \mathcal{R}$, define

$$\mathcal{R}^b_D(\sigma, \langle s, a, s' \rangle) = \left\{ s_1 \in \mathcal{S} \;\middle|\; \begin{array}{l} s_0 \in \sigma,\ \langle s_0, a, s_1 \rangle \in \mathcal{R}_D, \text{ and} \\ V(F^b(a,s), s_1) = V(F^b(a,s), s') \end{array} \right\}$$

$\mathcal{R}^b_D$ defines the belief state resulting from applying an action a to world state s, resulting in world state s', and initially not knowing in which of the world states in σ we are. If a includes sensing a fluent f in the resulting situation, then the resulting belief state will include only world states that agree with s' about the truth value of f.

2.3 Projection and Planning

In partially observable domains, plans need to branch on conditions that we can determine from the belief state. A plan in a transition system is the empty plan ϵ, an action $a \in \mathcal{A}$, the concatenation $\pi_1; \pi_2$ of two plans π_1, π_2, or the conditional plan $F \,?\, \pi_1 : \pi_2$ (read "if F then π_1, else π_2"), with F a propositional formula in $\mathcal{L}(\mathcal{P})$.

The following definition of projection assumes that we are given a set of pairs of the form ⟨world state, belief state⟩. The intuition is that, depending on the world state we may actually be in, we may be in a different belief state. Thus, in general, there may be several different belief states associated with a single world state, depending on the way we arrived at that world state (the initial state and the sequence of actions that we took). We use our transition operators $\mathcal{R}$ and $\mathcal{R}^b$ to define the resulting belief state from each action. When there is no transition in $\mathcal{R}$ for s, a, we end up in a distinguished state $Sink$ and an empty belief state (no state is possible). When our plan includes a condition (step 4), we decide on our action according to our knowledge. If we know the value of F, we follow the directive of the plan. Otherwise, we stay put (perform neither of the branches).

Definition 2.1 (Projection) *Let $B \subseteq (\mathcal{S} \cup \{Sink\}) \times \mathcal{S}^b$ be a nonempty set of pairs in which the first element is a possible world state or $Sink$ and the second is a belief state that we hold in this state. The projection of a plan π is defined as follows:*

1. $Proj[\epsilon](B) = B$;
2. $Proj[a](B) = \left\{ \langle s', \sigma' \rangle \;\middle|\; \begin{array}{l} \langle s, \sigma \rangle \in B,\ \mathcal{R}(\langle s, a, s' \rangle), \\ \mathcal{R}^b(\sigma, \langle s, a, s' \rangle) = \sigma' \end{array} \right\}$
$\bigcup \{ \langle Sink, \emptyset \rangle \mid \langle s, \sigma \rangle \in B,\ \forall s' \neg \mathcal{R}(\langle s, a, s' \rangle) \}$;
3. $Proj[\pi_1; \pi_2](B) = Proj[\pi_2](Proj[\pi_1](B))$;
4. $Proj[F \,?\, \pi_1 : \pi_2](B) =$
$Proj[\pi_1](\{\langle s, \sigma \rangle \in B \mid \forall s' \in \sigma\ F \text{ is true in } s'\}) \cup$
$Proj[\pi_2](\{\langle s, \sigma \rangle \in B \mid \forall s' \in \sigma\ F \text{ is false in } s'\}) \cup$
$\{\langle s, \sigma \rangle \in B \mid \exists s', s'' \in \sigma\ F \text{ is true in } s' \text{ and false in } s''\}$.

This definition of projection allows us to accumulate knowledge (encoded in the belief state associated with the world state we are in) and predict the state of our knowledge as a result of any sequence of actions, including sensing, whether the sequence of actions resulted from a conditional plan or not. The use of a $Sink$ state is similar to the distinction between *legal states* and illegal ones as in (Reiter 1995) (other form of this is the accessibility of a situation from $S0$ as in (Lin & Reiter 1994)).

Definition 2.2 (Planning Problem and Solution) *A planning problem is a tuple $\langle I, G, D \rangle$ in which D is a domain description, $I \subseteq \mathcal{S}$ is a non-empty initial set of world states and $G \subseteq \mathcal{S}$ is a set of goal world states. A solution for $\langle I, G, D \rangle$ is a plan π such that $Proj[\pi](I \times \{I\}) \subseteq G \times \mathcal{S}^b$.*

Thus, we assume that we start in a world state in I, but we do not know which world in I it is. The definition of the solution of a planning problem asserts that we wish to arrive at one of the goal states, regardless of the belief state we may be in at the end. When it is more convenient, we represent I using a set of value propositions ("**initially** F") and G using a propositional formula over $\mathcal{P}_D$. Notice that $\langle Sink, \emptyset\rangle \notin Proj[\pi](I \times \{I\})$ if $Proj[\pi](I \times \{I\}) \subseteq G \times \mathcal{S}^b$ because $Sink \notin \mathcal{S}$ and $G \subseteq \mathcal{S}$.

EXAMPLE Assume that we are given a blocks world domain in which we do not know if A is on B or vice versa, and A, B are the only blocks. Let $on(A, B)$ be the fluent that says that A is on B and $on(B, A)$ the fluent saying that B is on A. $I = \{\{on(A, B)\}, \{on(B, A)\}\}$. Now, assume that our goal is to make sure that B is on A, and we have full observability once the plan is executed (e.g., we have a rule "**observed** $on(A, B)$" and a rule "**observed** $on(B, A)$"). Then, we can form the plan $\langle nullAction;\ on(B, A)\ ?\ nullAction\ :\ (putOnTable(A);\ putOn(B, A))\rangle$. Executing the first step in this plan from the state of knowledge I results in one of the states of knowledge $\{\{on(A, B)\}\}$ (in the world state in which $on(A, B)$ holds) and $\{\{on(B, A)\}\}$ (in the world state in which $on(B, A)$ holds). The condition can then be evaluated and the proper set of actions taken. ■

We choose to require an action before observations can be made because this seems more natural to us (the robot *wakes up* to find itself in this state).

3 Complexity of Nondeterministic Planning

In the following, the *poly-plan-size planning problem* is the problem of finding plans that can be represented in polynomial space, and the *poly-depth planning problem* is the problem of finding plans that execute in polynomial time. When we talk about the *planning problem*, we refer to both. For lack of space, we refrain from discussing computational complexity theory, and the reader is referred to (Papadimitriou 1994). However, we remind the reader that the following relationships hold between complexity classes:

$$P \subseteq \begin{matrix}\Sigma_1^P \\ \Pi_1^P\end{matrix} \subseteq \Delta_2^P \subseteq \begin{matrix}\Sigma_2^P \\ \Pi_2^P\end{matrix} \subseteq \ldots \subseteq PH \subseteq NP^{PP} \subseteq PSPACE$$

where NP=Σ_1^P, coNP=Π_1^P, the characteristic Σ_i^P problem is $\exists x_i \forall x_{i-1} \exists ... \varphi$ and the characteristic Π_i^P problem is $\forall x_i \exists x_{i-1} \forall ... \varphi$, with φ a propositional formula with variables $x_1, ..., x_i$ (for example, the characteristic NP problem is $\exists x \varphi$). Also, $PH = \bigcup_{i \geq 1} \Sigma_i^P$.

3.1 Conformant Planning (no Observations)

In conformant planning with nondeterministic actions (e.g., (Bertoli, Cimatti, & Roveri 2001)) there is an incompletely known initial state, and no observability. Consequently, planning includes no sensing actions, no automatically sensed conditions, and no branching on conditions.

Our model of projection (Definition 2.1) can be simplified for this scenario. In particular, all the belief states associated with world states in our definition of projection and planning are identical. To see this, notice that $F^b(a, s) = \emptyset$ for world state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$ because there are no sensing actions or observations. Thus, $\mathcal{R}_D^b(\sigma, \langle s, a, s'\rangle) = \mathcal{R}_D^b(\sigma, \langle t, a, t'\rangle)$ for every $s, t, s', t' \in \mathcal{S}$ such that $\langle s, a, s'\rangle, \langle t, a, t'\rangle \in \mathcal{R}$. As a result, for a planning problem, we can represent the state after projection as a single belief state, which may be compactly representable using a set of propositional formulae on the state fluents. This allows us to prove the following.

Theorem 3.1 *For planning problems with incomplete information about the initial state, nondeterministic actions, and no sensing, the planning problem is Σ_2^P-complete. If deciding whether an action a is executable in s is NP-complete, then then the problem is Σ_3^P-complete.*

A similar theorem has been established for a different nondeterministic action language in (Eiter *et al.* 2000) for a different action language. Examples of actions for which executability can be checked in polynomial-time in a given state are actions with no nondeterministic effects, or effects that are a disjunction of at most two literals. In those cases, satisfiability of $F(a, s)$ can be checked in linear time.

PROOF SKETCH The proof is similar to that of Theorem 3.4, and we omit it for lack of space. The main difference is that we use the assumption of no observability when we check how much time it takes us to decide executability of an action a in the plan π in state s that resulted from a partial execution of π from u. Since s is fully given by x, checking executability of a can be done in polynomial time. The execution of π, x is a simple deterministic projection operation. Thus, checking $\exists \pi \forall u, x\ P(\pi, u, x, w)$ is in Σ_2^P. ■

A related interesting result in this context is the complexity of answering a query about the result of projection.

Theorem 3.2 *Let D be a planning domain with nondeterministic actions and no sensing. The problem of answering whether all the states in $Proj[\pi](I \times \{I\})$ satisfy Q, for a plan π and a query, Q, is coNP-complete.*

The proof is similar to that given in (Baral, Kreinovich, & Trejo 2000) for the deterministic case.

3.2 Planning and Full (Automatic) Observability

In planning under full automatic observability (e.g., (Cimatti, Roveri, & Traverso 1998)) we assume nondeterministic actions and incompletely specified initial state, and we allow conditioned branches in the plan. No action needs to be performed to retrieve the information, and the sensing is done consistently for every time step.

Under full automatic observability, the belief state associated with every world state in our projection $Proj$ (Definition 2.1) includes only that world state. Thus, when we execute a condition step in a conditional plan (part 4 of Definition 2.1), we decide on the subsequent sub-plan only according to the current state of the world.

Theorem 3.3 *For planning problems with incomplete information about the initial state, with nondeterministic action, and with full automatic observability, the poly-depth planning problem is PSPACE-complete.*

PROOF SKETCH The proof is very similar to the one given in (Littman 1997) for probabilistic planning with full observability, and proceeds by reduction from quantified boolean formulae (QBF) (validity of formulae of the form $\exists x_1 \forall y_1 \exists x_2 \forall ... \exists x_k \forall y_k \Phi$). Briefly, every existentially-quantified variable is a plan step, and every universally-quantified variable is a nondeterministic choice made by the world. The result of this nondeterministic choice is At the end, we put an action **evaluate-formula**, which leads to the successful goal (SAT) iff the conditions (the clauses of Φ) are satisfied by the selected actions. ■

Theorem 3.4 *For planning problems with incomplete information about the initial state, nondeterministic actions, and full automatic observability, the poly-plan-size planning problem is Σ_2^P-complete.*

PROOF First, we show that our problem is in Σ_2^P, assuming that either we do not check executability of the plan, or that checking executability can be done in polynomial time. The existence of a successful plan means the existence of a plan π such that for every set of values u for the fluents in the initial state, if u satisfies the conditions known for the initial state, then any execution of the plan leads to a goal state. To check that the plan is a legal/executable one (the actions sanctioned by the plan are executable when we execute them) we need to verify that there is a state in the result of every plan action to every branch of the plan. If this involves an NP-hard computation, then our problem is in Σ_3^P. First, we assume every action is executable in every state (e.g., having no effect if it is not executable).

Mathematically, the existence of a plan can be written as a formula $\exists \pi \forall u, x \ P(\pi, u, x, w)$, where the predicate $P(\pi, u, x, w)$ describes the fact that for the planning problem w and for the values u of the initial fluents, the execution x of plan π leads to a goal state, if u satisfies the conditions of the initial state and x chooses the nondeterministic results of actions according to the effect propositions. Put differently, x is an encoding of $|\pi|$-many steps (the states (fluent values) in the chosen execution sequence), and P verifies that if x is a valid sequence of steps of π, starting from state u (and u is a valid initial state), then the execution of π satisfies the goal.

To prove that this problem belongs to Σ_2^P, we must show that the quantifiers run over variables of tractable length, and that $P(\pi, u, x, w)$ is computable in polynomial time.

The variable π runs over plans and is, therefore, tractable: The number of variables in the vector π is bounded by the size of the plans that we consider. Since we consider only polynomially-long plans, there are only polynomially-many variables in this vector (where each variable takes $|\mathcal{A}|$ values). The variable u runs over sets of values of fluents for the initial state. Viewed as a vector of propositional variables this is a vector of size n, where n is the number of propositional fluents in $\mathcal{P}$. Thus, u is of tractable length. The variable x runs over the set of nondeterministic choice points in π. For each nondeterministic choice of a world, the choice is specified by a nondeterministic choice of values for the fluents in $\mathcal{P}$. Thus, since π is of polynomial length, x is of polynomial length as well.

If we know the values of π, u, x, w, then $P(\pi, u, x, w)$ needs to check that u satisfies the initial conditions (linear time in the encoding of the sentences "**initially** F"), that for every point in executing π, x satisfies the conditions of the applicable effect propositions (x is an encoding of $|\pi|$-many states, so the computation, for every state, that the chosen resulting state satisfies the effect propositions takes polynomial time in the encoding of the sentences "a **causes** F **if** G"), and that the execution of π, x satisfies w (linear time in the encoding of the propositional goal sentence/s).

Full observability comes in when we need to evaluate a condition in our plan to decide on the action that will be taken in our current state. Given full observability, our knowledge state at each step of the execution (possibly besides the initial knowledge state) includes exactly one state. Since this is at most one more condition to be checked in every step, and this check is done in polynomial time *given a state*, full observability leaves P polynomial-time checkable. The problem involved with checking a condition in the initial state can be sidesteped by requiring the first action to be a null action (leaving us in the same state).

The execution of π, x is a simple deterministic projection operation. Thus, checking $\exists \pi \forall u, x \ P(\pi, u, x, w)$ is in Σ_2^P, if we can check that a is executable in polynomial time.

Now we show that this problem is Σ_2^P-hard. Let F be any quantified boolean formula (QBF) of the form $\exists x_1, ..., x_n \forall y_1, ..., y_m \Phi$, where Φ consists of t clauses. We construct a problem instance P and show that F is true iff P has a solution. Define $P = \langle I, G, D \rangle$ where $G = \{sat\}$, and

$$I = \left\{ \begin{array}{l} \textbf{initially } \neg x_1 \wedge ... \wedge \neg x_n, \\ \textbf{initially } s_0 \wedge \neg s_1 \wedge \neg s_2 \wedge ... \wedge \neg s_{n+2} \end{array} \right\}$$

$$D = \left\{ \begin{array}{l} \left. \begin{array}{l} a \textbf{ causes } s_i \wedge x_i \wedge \neg s_{i-1} \textbf{ if } s_{i-1} \\ a\text{- } \textbf{causes } s_i \wedge \neg x_i \wedge \neg s_{i-1} \textbf{ if } s_{i-1} \end{array} \right\} i \leq n \\ a \textbf{ causes } (y_1 \vee \neg y_1) \wedge ... \wedge (y_m \vee \neg y_m) \wedge \\ \qquad\qquad s_{n+1} \wedge \neg s_n \textbf{ if } s_n \\ a \textbf{ causes } sat \wedge s_{n+2} \wedge \neg s_{n+1} \textbf{ if } \Phi \wedge s_{n+1} \\ \textbf{observed } x_i \quad (i \leq n) \\ \textbf{observed } y_i \quad (i \leq m) \\ \textbf{observed } s_i \quad (i \leq n+2) \end{array} \right\}$$

This planning problem has a solution iff $\exists x_1, ..., x_n \forall y_1, ..., y_m \Phi$ is a valid formula. Also, every successful plan is of length at most $n+2$, because we keep advancing the time counter $(s_0, ..., s_{n+2})$ with every action performed, and there is no way to make sat true after the $n+2$-th step. Thus, our planning problem is at least as hard as $\exists \forall$ QBFs, meaning that it is Σ_2^P-hard.

Finally, if checking that an action a is executable in a state s is NP-complete (depending on our effect propositions, it may involve checking that there is a state that satisfies the consequences of our effect propositions), then the problem is in Σ_3^P by the same argument as before, having the computation of $P(\pi, u, x, w)$ NP-complete.

For the general case of checking that actions are executable, we need to show that every $\exists \forall \exists$ QBF, $\exists x_1, ..., x_n \forall y_1, ..., y_m \exists z_1, ..., z_l \Phi$ can be reduced to such a

planning problem. The construction is as above:

$$I = \left\{ \begin{array}{l} \textbf{initially } \neg x_1 \wedge ... \wedge \neg x_n \wedge \neg z_1 \wedge ... \wedge z_l \\ \textbf{initially } s_0 \wedge \neg s_1 \wedge \neg s_2 \wedge ... \wedge \neg s_{n+l+2} \end{array} \right\}$$

$$D = \left\{ \begin{array}{l} \left. \begin{array}{l} a \textbf{ causes } s_i \wedge x_i \wedge \neg s_{i-1} \textbf{ if } s_{i-1} \\ a\textbf{- causes } s_i \wedge \neg x_i \wedge \neg s_{i-1} \textbf{ if } s_{i-1} \end{array} \right\} i \leq n \\ a \textbf{ causes } (y_1 \vee \neg y_1) \wedge ... \wedge (y_m \vee \neg y_m) \wedge \\ \qquad s_{n+1} \wedge \neg s_n \textbf{ if } s_n \\ \left. \begin{array}{l} a \textbf{ causes } s_{n+i+1} \wedge z_i \wedge \neg s_{n+i} \textbf{ if } s_{n+i} \\ a\textbf{- causes } s_{n+i+1} \wedge \neg z_i \wedge \neg s_{n+i} \textbf{ if } s_{n+i} \end{array} \right\} i \leq l \\ a \textbf{ causes } sat \wedge s_{n+l+2} \wedge \neg s_{n+l+1} \textbf{ if } \Phi \wedge s_{n+l+1} \\ \textbf{observed } x_i \quad (i \leq n) \\ \textbf{observed } y_i \quad (i \leq m) \\ \textbf{observed } z_i \quad (i \leq l) \\ \textbf{observed } s_i \quad (i \leq n+l+2) \end{array} \right\}$$

It follows that our problem is then Σ_3^P-complete. ■

3.3 Planning and Partial Observability

For planning with partial automatic observability (e.g., (Bertoli *et al.* 2001)) we assume that there are propositional properties of the current state that are always observed and that these are the only observations made. We allow conditional branches in the plan, nondeterministic actions and incomplete knowledge about the initial state.

We consider two cases. In the first, we assume that conditions for branching of the plan can be made only on one of the observable propositional formulae F declared "**observed** F" (this is the approach taken by (Bertoli *et al.* 2001)). In the second, we assume that conditions for branching of the plan can be made on any formula on $\mathcal{P}$.

In the first case, our model of projection (Definition 2.1) can be simplified in a fashion similar to the one we used for planning with full observability (Section 3.2). The key observation is that in this case there is no need to *memorize* or *deduce* the value of F because our plan can branch on the observation as it occurs. Thus, it is not necessary to include a separate belief state for every possible trajectory of our plan. Instead, we treat the world as if it is fully observable (it is, as far as branching on conditions is concerned). Consequently, we get identical results to those for full observability (Σ_2^P-completeness for poly-plan-size planning without executability checking, Σ_3^P-completeness if checking executability is NP-complete, and PSPACE-completeness for poly-depth planning).

For general partially observable planning, we prove the following result.

Theorem 3.5 *Let D be a planning domain with nondeterministic actions and partial observability. Let π be a plan and Q a query. The problem of answering whether all the states in $Proj[\pi](I \times \{I\})$, that result from the actual execution $s_1, ..., s_n$ of π, satisfy Q, is Π_2^P-complete.*

PROOF Assume first that the decision of whether a is executable in a state s can be done in polynomial time. To see that the problem is in coNP, notice first that there is only one belief state (a set of world states) that results from the actual execution of π that yields a sequence $s_1, ..., s_n$. Let us call this belief state σ. In this form, the problem we need to answer is $\sigma \models Q$ ("$\models$" is the logical entailment relation).

Instead of trying to compute σ, we can write the problem in QBF format as $\forall u, x\, P(\pi, s_1, ..., s_n, u, x, w)$, with u being any possible initial state (w includes the initial conditions, I), x any possible execution of the nondeterministic plan π that follows the update rule of $\mathcal{R}_D^b$ for $s_1, ..., s_n$, and P checking that this hypothetical execution x starting from hypothetical initial state u arrives at a state that satisfies Q.

At every step of the execution, P needs to check if the relationship between the current belief state and a possible condition in the plan, deciding on the action accordingly. Given the actions decided on so far, the belief state is the set of states that may result from this execution and that are consistent with the observations. Thus, the decision in step i by $P(\pi, s_1, ..., s_n, u, x, w)$ can be written as

$$\begin{array}{l} (\forall s\ R_i(\pi, s_1, ..., s_i, s) \Rightarrow Cond_i(s)) \Rightarrow \\ \quad Resulting_T(\pi, i, s_i, s_{i+1}) \wedge \\ \quad (\forall s'\ (R_{i+1}(s_1, ..., s_i, s_{i+1}, s') \Leftrightarrow \\ \qquad \exists s\ R_i(s_1, ..., s_i, s) \wedge \\ \qquad Resulting_T(\pi, i, s, s') \wedge \\ \qquad V(F^b(\pi, i+1, s_1, ..., s_i), s') = \\ \qquad\quad V(F^b(\pi, i+1, s_1, ..., s_i), s_{i+1})) \end{array}$$

when $V(\varphi, s)$ is defined as in Section 2.2 and $F(\pi, i+1, s_1, ..., s_i)$ is $F(a, s_i)$ (as defined in Section 2.2) for a being the action taken in step $i+1$ of π after the sequence of states $s_1, ..., s_i$, and R_i corresponds to the accessibility relation in step i of the plan execution ($R_i(\pi, s_1, ..., s_i, s)$ means that s is a state in our belief state at step i).

Informally, this formula says that if every state that is in our belief state at state s_i satisfies the condition posed in step $i+1$, then s_{i+1} is a possible state resulting from applying the next action in π (otherwise we would choose a different action and this state may not be in the result of that action), and a state is in the belief state at that resulting point iff it is the result of the same action from one of the previously possible states and it agrees with the sensory information found in state s_{i+1}.

As is, this formula is in Π_2^P because it is of the form $\forall s'\ (\varphi(s') \Leftrightarrow \exists s\ \psi(s', s))$, which is equivalent to $\forall s' \forall s_1 \exists s_2\ ((\varphi(s') \vee \neg\psi(s', s_1)) \wedge (\neg\varphi(s') \vee \psi(s', s_2)))$. Thus, our problem is in Π_2^P in this case (not checking executability of actions) because P is equivalent a polynomial-length formula of this form (duplicating this formula for all the steps of the plan).

To see that the general problem is also Π_2^P-complete (the general problem includes that of checking there is a state in the result of applying an action) notice that for checking executability we need to add the condition $(\forall s\ R_i(\pi, s_1, ..., s_i, s) \Rightarrow Cond_i(s)) \Rightarrow \exists s_{i+1} Resulting_T(\pi, i, s_i, s_{i+1})$. This condition is in Π_2^P. Thus, letting P be the combination of the two rules for all the steps of the plan leads to P being in Π_2^P.

The proof of completeness for Π_2^P for the restricted and general problems, respectively, follows from the similar completeness results for nondeterministic actions without observations (Theorem 3.2). ■

As a consequence, we can prove the following.

Theorem 3.6 *For planning problems with incomplete information about the initial state, with nondeterministic actions, and with partial automatic observability, the poly-plan-size planning problem is Σ_3^P-complete.*

PROOF SKETCH The proof is similar to the proof of Theorem 3.4, using the proof of Theorem 3.5 as a skeleton for showing that the problem is in Σ_3^P. Then, completeness follows from Theorem 3.4.

Theorem 3.7 *For planning problems with incomplete information about the initial state, with nondeterministic actions, and with partial automatic observability, the poly-time planning problem is PSPACE-complete.*

The proof is similar to that of Theorem 3.3.

3.4 Planning and Sensing Actions

Planning with sensing actions is very closely related to planning in partially-observable domains with automatic sensing. The presumed difficulty with sensing actions is that now we need to maneuver the executing agent to a state where it can sense features of the state that it needs. Despite this additional difficulty, the problem has similar complexity to its automatic-sensing version.

Theorem 3.8 *For planning problems with incomplete information about the initial state, with nondeterministic actions, and with sensing actions, the poly-plan-size planning problem is Σ_3^P-complete.*

Theorem 3.9 *For planning problems with incomplete information about the initial state, with nondeterministic actions, and with sensing actions, the poly-depth planning problem is PSPACE-complete.*

The proofs of both theorems are similar to those of Theorems 3.6, 3.3, respectively.

4 Results for Other Specification Languages and Semantics

There are several other semantics that have been used for specifying the effects of nondeterministic actions. Some of the prominent ones are: (1) GOLOG (Reiter 2001) includes a nondeterministic choice between deterministic actions (and nondeterministic choice between arguments that specify deterministic actions); (2) (Haslum & Jonsson 1999) specify a language that allows nondeterministic choice between actions, similar to that used for GOLOG; (3) $\mathcal{AR}$ (Giunchiglia, Kartha, & Lifschitz 1997) specifies nondeterministic actions with the same language as we do, but uses a minimal-change semantics for all fluents; and (4) $\mathcal{C}$ (Giunchiglia & Lifschitz 1998; Ferraris & Giunchiglia 2000) includes control over inertia for different fluents, given actions and state conditions. This allows it to specify nondeterminism of the kind we present and of the kind of $\mathcal{AR}$.

Generally speaking, our proofs apply to these semantics as well, with some modification. The clearest connection is with the nondeterministic choice of actions as given in GOLOG and (Haslum & Jonsson 1999). Those can be reduced to our semantics with only polynomial growth in size, providing identical results for those systems.

The minimization of change policies, applied by $\mathcal{AR}$ and $\mathcal{C}$, are typical of Π_2^P-complete problems (Eiter & Gottlob 1993). This suggests an increase in the complexity of planning by one complexity class in the polynomial hierarchy. We hope to report on this in the final version of this paper.

5 Related Work

It is interesting to portray our work in the context of other results achieved for deterministic actions with incomplete knowledge about the initial state, and compared to results obtained for probabilistic systems. We display those in Table 1. For probabilistic planning we assume that we wish to find a finite-horizon policy that is time dependent (this is the closest problem to classical AI planning).

It is interesting to contrast these results with the results for the unbounded-plan-size problems. Those are PSPACE-complete for deterministic planning (Bylander 1994), EXPSPACE-complete for unobservable, incomplete-initial-state, deterministic planning (Haslum & Jonsson 1999), EXPTIME-completeness for MDP planning (fully observable) (Littman 1997).

6 Conclusions

We have provided a formal model of reasoning and planning in the presence of nondeterministic actions, incomplete knowledge about the initial state and different scenarios of observability. We also presented a language and semantics for specifying such planning domains. Using this model we presented a set of computational complexity results that are surprising in their closeness to planning with deterministic actions. These results suggest that it may be worthwhile to try to approximate probabilistic planning tasks using planning with nondeterministic effects. It also shows that some of the problems that we analyzed are well-suited for encoding in propositional representations, and may allow relatively-efficient solutions, along the lines already explored for planning with deterministic actions and some nondeterministic planning domains.

There are several places where the language and results should be extended. In particular, the action language should be extended to allow concurrent actions, ramifications or qualification constraints. Also, the observation model is deterministic (we know the actual answer to our *query* F), and the language should be extended to allow observations of one of F or G (which is different from observing $F \vee G$). We hope to do these in the near future.

7 Acknowledgments

I wish to thank Mark Paskin for comments on a draft of this paper. This research was supported by a National Science Foundation grant ECS-9873474.

Planning Scenario (polynomial plan-size)	Deterministic Effects	Nondeterministic Effects	Probabilistic Effects
No observations; Complete initial state	NP-complete [a],[b]	Σ_3^P/Σ_2^P-complete (§3.1)	NP^{PP}-complete [f]
No observations; Incomplete initial state	Σ_2^P-complete [a]	Σ_3^P/Σ_2^P-complete (§3.1)	NP^{PP}-complete [f]
Full observability; Incomplete initial state	in Δ_2^P [a]+[d](*)	Σ_3^P/Σ_2^P-complete (§3.2)	NP^{PP}-complete [c]
Partial observability; Incomp. initial state	in Σ_3^P [a]+(§3.3)	Σ_3^P-complete (§3.3)	??
Sensing actions; Incomplete initial state	in Σ_3^P [a]+(§3.4)	Σ_3^P-complete (§3.4)	??

Planning Scenario (polynomial depth)	Deterministic Effects	Nondeterministic Effects	Probabilistic Effects
No observations; Complete initial state	NP-complete [a],[b]	Σ_3^P/Σ_2^P-complete (§3.1)	NP^{PP}-complete [f]
No observations; Incomplete initial state	Σ_2^P-complete [a]	Σ_3^P/Σ_2^P-complete (§3.1)	NP^{PP}-complete [f]
Full observability; Incomplete initial state	Π_2^P-complete [a],[d]	PSPACE-complete (§3.2)	PSPACE-complete [e]
Partial observability; Incomp. initial state	PSPACE-complete [a]	PSPACE-complete (§3.3)	??
Sensing actions; Incomplete initial state	PSPACE-complete [a]	PSPACE-complete (§3.4)	??

Table 1: Complexity classes for sequential planning problems (input encoded using state propositions). [a]=(Baral, Kreinovich, & Trejo 2000); [b]=(Bylander 1994; Kutluhan Erol & Subrahmanian 1995); [c]=(Littman 1997); [d]=(Rintanen 1999); [e]=(Littman, Goldsmith, & Mundhenk 1998); [f]=(Mundhenk, Goldsmith, & Allender 1997). (*) [d] showed inclusion in Σ_2^P, and [a] showed inclusion in Π_2^P for planning with polynomial depth. We write Σ_3^P/Σ_2^P to indicate the two cases of checking executability of actions (corresponding to *improper* and *proper* planning domains, respectively, in (Eiter *et al.* 2000)).

References

Baral, C.; Kreinovich, V.; and Trejo, R. 2000. Computational complexity of planning and approximate planning in the presence of incompleteness. *Artificial Intelligence* 122(1-2):241–267.

Bertoli, P.; Cimatti, A.; Roveri, M.; and Traverso, P. 2001. Planning in nondeterministic domains under partial observability via symbolic model checking. In *IJCAI '01*, 473–478. MK.

Bertoli, P.; Cimatti, A.; and Roveri, M. 2001. Heuristic search + symbolic model checking = efficient conformant planning. In *IJCAI '01*, 467–472. MK.

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *Artificial Intelligence* 69(1–2):165–204.

Cimatti, A.; Roveri, M.; and Traverso, P. 1998. Automatic OBDD-based generation of universal plans in non-deterministic domains. In *Proc. AAAI '98*, 875–881.

Dean, T., and Kanazawa, K. 1988. Probabilistic temporal reasoning. In *Proc. AAAI '88*, 524–528.

Doucet, A.; de Freitas, N.; Murphy, K.; and Russell, S. 2000. Rao-Blackwellised particle filtering for dynamic bayesian networks. In *Proc. UAI '00*, 176–183. MK.

Eiter, T., and Gottlob, G. 1993. Propositional circumscription and extended closed-world reasoning are pi /sub 2//sup P/-complete. *Theoretical Computer Science* 114(2):231–245.

Eiter, T.; Faber, W.; Leone, N.; Pfeifer, G.; and Polleres, A. 2000. Planning under incomplete knowledge. In *Proceedings of Computational Logic*.

Ferraris, P., and Giunchiglia, E. 2000. Planning as satisfiability in nondeterministic domains. In *Proc. AAAI '00*, 748–753.

Giunchiglia, E., and Lifschitz, V. 1998. An action language based on causal explanation: preliminary report. In *Proc. AAAI '98*, 623–630.

Giunchiglia, E.; Kartha, G. N.; and Lifschitz, V. 1997. Representing Action: Indeterminacy and Ramifications. *Artificial Intelligence*. to appear.

Giunchiglia, E. 2000. Planning as satisfiability with expressive action languages: concurrency, constraints and nondeterminism. In *Proc. KR '2000*, 657–666. MK.

Haslum, P., and Jonsson, P. 1999. Some results on the complexity of planning with incomplete information. In *European Conference on Planning (ECP-99)*, 308–318. Springer-Verlag.

Kutluhan Erol, D. S. N., and Subrahmanian, V. 1995. Complexity, decidability and undecidability results for domain-independent planning. *Artificial Intelligence* 76(1–2):75–88.

Lin, F., and Reiter, R. 1994. State constraints revisited. *Journal of Logic and Computation* 4(5):655–678.

Littman, M. L.; Goldsmith, J.; and Mundhenk, M. 1998. The computational complexity of probabilistic planning. *Journal of AI Research* 9:1–36.

Littman, M. L. 1997. Probabilistic propositional planning: Representations and complexity. In *Proc. AAAI '97*, 748–761.

McCarthy, J., and Hayes, P. J. 1969. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In Meltzer, B., and Michie, D., eds., *Machine Intelligence 4*. Edinburgh University Press. 463–502.

Mundhenk, M.; Goldsmith, J.; and Allender, E. 1997. The complexity of policy evaluation for finite-horizon partially-observable Markov decision processes. In *22nd Int'l Symboposium on Mathematical Foundations of Computer Science (MFCS'97)*.

Papadimitriou, C. H. 1994. *Computational Complexity*. Addison-Wesley.

Reiter, R. 1995. On specifying database updates. *Journal of Logic Programming* 25(1):53–91.

Reiter, R. 2001. *Knowledge In Action: Logical Foundations for Describing and Implementing Dynamical Systems*. MIT Press.

Rintanen, J. 1999. Constructing confitional plans by a theorem-prover. *Journal of AI Research* 10:323–352.

Son, T. C., and Baral, C. 2001. Formalizing sensing actions a transition function based approach. *Artificial Intelligence* 125(1–2):19–91.

Turner, H. 2001. Polynomial-length planning spans the polynomial hierarchy. Unpublished note.